6-2005

Semantic Management of Web Services using the Core Ontology of Services

Daniel Oberle

Steffen Lamparter

Andreas Eberhart

Stephan Grimm

Sudhir Agarwal

See next page for additional authors

Semantic Management of Web Services using the Core Ontology of Services

Daniel Oberle [1*], Steffen Lamparter [1†], Andreas Eberhart [2], Steffen Staab [3,4], Stephan Grimm [5‡], Pascal Hitzler [1§], Sudhir Agarwal [1¶], Rudi Studer [1,4,5]

[1] AIFB, Universität Karlsruhe, Germany, lastname@aifb.uni-karlsruhe.de

[2] Hewlett-Packard, Waldorf, Germany, andreas.eberhart@hp.com

[3] ISWeb, University of Koblenz-Landau, Germany, staab@uni-koblenz.de

[4] Ontoprise GmbH, Karlsruhe, Germany

[5] FZI Research Center Karsruhe, Gemany

Introduction

Different Web Service standards like WSDL, WS-Security, WS-Policy etc., henceforth referred to as WS*, factorize Web Service management tasks into different aspects, such as input/output, workflow, or security. The advantages of WS* are multiple and have already achieved industrial impact. WS* descriptions are exchangeable and developers may use different implementations for the same Web Service description. The disadvantages of WS*, however, are also apparent: even though the different standards are complementary, they must overlap and one may produce models composed of different WS* descriptions, which are inconsistent with each other, but the reasons for the inconsistencies are not easily determined. This is the case because there is no coherent conceptual model of WS*, i.e. terms with equivalent semantics are introduced differently in the respective XML-DTDs. Thus, it is impossible to ask for conclusions that come from integrating different WS* descriptions. Hence, discovering such Web Service management problems or asking for other similar kinds of conclusions that derive from the integration of WS* descriptions remains a purely manual task to be done by the software developers accompanied by little to no formal machinery.

Researchers investigating Semantic Web Services have clearly articulated these shortcomings of WS* standardization and have been presenting approaches to counter some of them [6]. The core of their proposals lies in creating semantic standards, their principal objective being a far-reaching formalization that allows for full automation of Web Service management tasks such as discovery and composition. Again, the potential advantages are obvious; the disadvantages, however, are also apparent: It is unclear, what kind of powerful machinery could constitute a semantic model allowing for full automation, and indeed such full automation appears to be outside the scope of real-world software applications within the foreseeable future.

Therefore, we postulate that semantic management of Web Services should not try to tackle full automation of all Web Service management tasks as its objective, as this requires an understanding of the world that is too deep to be modelled explicitly. Instead, we foresee a more passive role for semantic management of Web Services. One that is driven by the needs of the developers, who must cope with the complexity of Web Service integration and WS* descriptions. They could use valuable tools for integrating previously separated aspects. As such, semantic management of Web Services has also been used as an example for the W3C Note of the Software Engineering Task Force.[1]

The kind of objectives that are to be approached by this semantic management are constrained by a trade-off between investing efforts for managing Web Services and investing efforts for semantic modelling of Web Services. The tradeoff is depicted qualitatively in Figure 1. The objective of full automation by semantic modelling will need very fine-grained, detailed modelling of all aspects of Web Services -- essentially everything that an intelligent human agent must know. Thus, modelling efforts skyrocket at the end of fine-grained modelling. At the other end, where modelling is very coarse and little modelling facilitates management, efforts for managing distributed systems soar as experiences have shown in the past. No matter what the exact scale of granularity and efforts are, the qualitative indication of management and modelling costs such as done in Figure 1 leads to an overall total cost picture as indicated in the same figure.

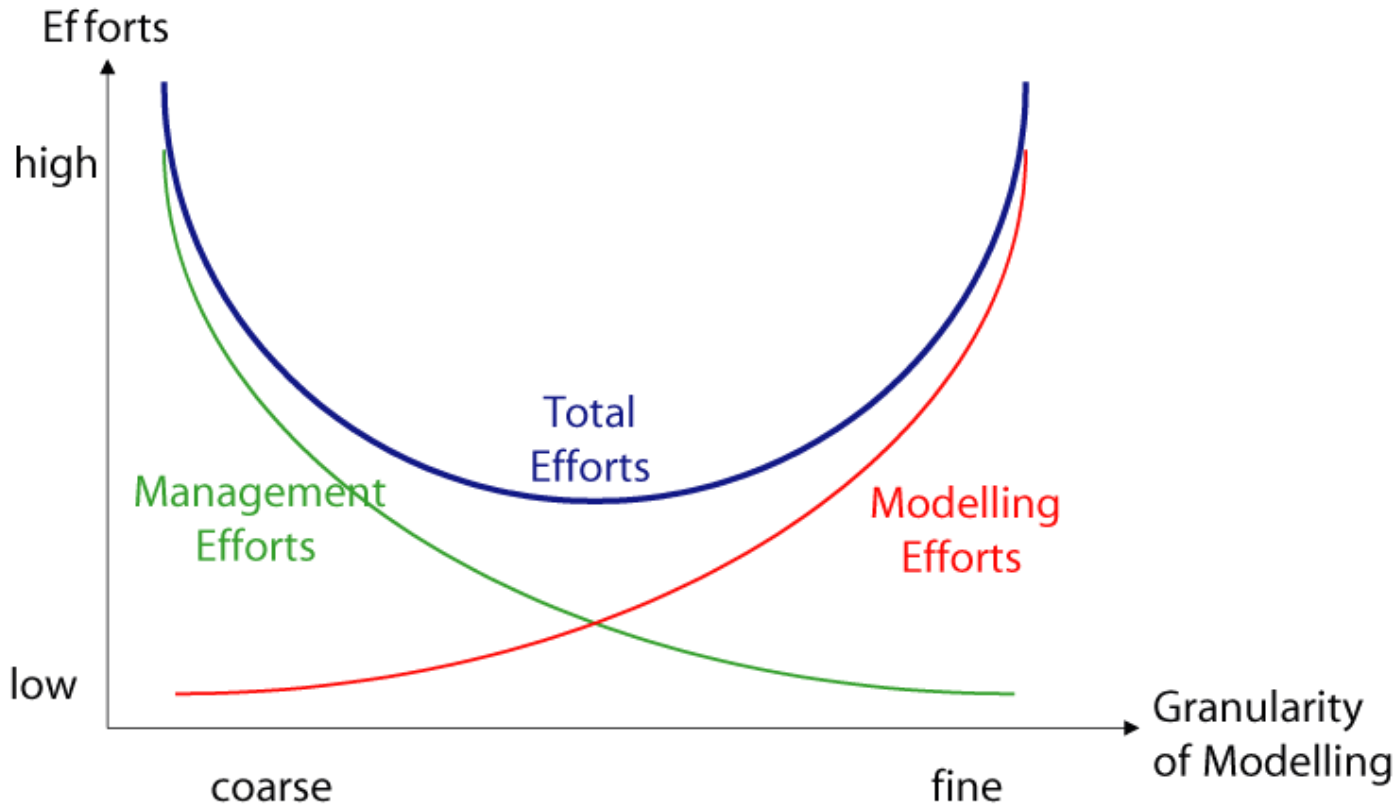

Figure 1: Cost of modelling vs. management

We realize our approach by means of a comprehensive ontology for modelling services and service management tasks, the Core Ontology of Services (CoS), based on the foundational ontology DOLCE [11]. We do not envision CoS as a standard, but understand it as a clean reference ontology for disambiguating overloaded terms like Web Service.

The remainder of this paper consists of sections containing a discussion of use cases, some remarks about CoS, a prototype description, related work, and conclusons.

Potential Use Cases

In order to take advantage from the trade-off depicted in Figure 1, potential application cases have to approach the point of minimal total efforts. We have identified and analysed a number of use cases which can be expected to gain from semantic management using our approach. They also allow us to distinguish semantic management from other tasks involving Web services development and from full automation like postulated by Semantic Web Services. A detailed account of our analysis is given in [9], and we restrict ourselves to the brief discussion of a few examples.

Detecting Loops in the Invocation Chain

Web Services based applications usually make use of asynchronous messaging, thus giving rise to complex interaction protocols between business partners. Current workflow design workbenches only visualize the local flow and leave the orchestration of messages with the business partners up to the developer. We believe that enough information is available in machine-readable format such that a tool can assist the developer in assessing the global flow. For instance, the structure of the local flow can be combined with publicly available abstract flows of the partners in order to detect loops in the invocation chain that would lead to non-termination of the system.

Incompatible Inputs and Outputs

Type checking is not as straightforward anymore using loosely coupled services operated by a large number of organizations. Furthermore, the interpretation of a B2B term such as `price' might be different, even though syntactically it refers to an agreed-upon XML Schema type. For instance, different partners might have different assumptions about the currency and taxation details. A system, which automatically compares communication inputs and outputs according to a more detailed model, will help to prevent unexpected behavior in a system.

Change Management

A system no longer being under the tight control of a single organizational unit will definitely be prone to service versioning issues. Updating a single component already requires close cooperation between the parties involved and this will without a doubt be much harder in Web Services based applications. Consequently, the respective process composition tool suite should provide support for monitoring the providers' service interface definitions.

Aggregating Service Information

Services will often be implemented based on other services. A service provider publishes information about its service. This might include service-level agreements indicating a guaranteed worst-case response time, the cost of the service, or average availability figures. The service requestor, in this case a composite service under development, can collect this information from the respective service providers. In turn, it offers a service and needs to publish similar numbers. We envision a tool that supports the administrator in providing a first cut of this data by aggregating the data gathered from external providers.

Further use cases

which have been detailed in [9] include the analysis of message contexts, the selction of service functionality and policies, the relating of communication parameters, and quality of service analysis.

The Core Ontology of Services

The use cases just discussed show that we want to be able to facilitate management tasks covering a broad range of aspects. The underlying ontology thus has to allow for the modelling of such diverse aspects as service profiles, service taxonomies, policies, workflow information, interface descriptions and quality of service information. We also aim at harmonizing the implicit, yet sufficiently similar, conceptual models that underly all the WS* efforts.

Besides the purpose of meeting the representation requirements, our aim is to lower the modelling costs as much as possible. In [7,4] we analyzed existing ontology efforts and came to the conclusion that their lack of ontology quality complicates the modelling task for several reasons. Hence, we try to simplify this task by re-using a proven foundational ontology instead of modelling everything from scratch. Foundational ontologies are high-quality formalizations of domain independent concepts and

associations.

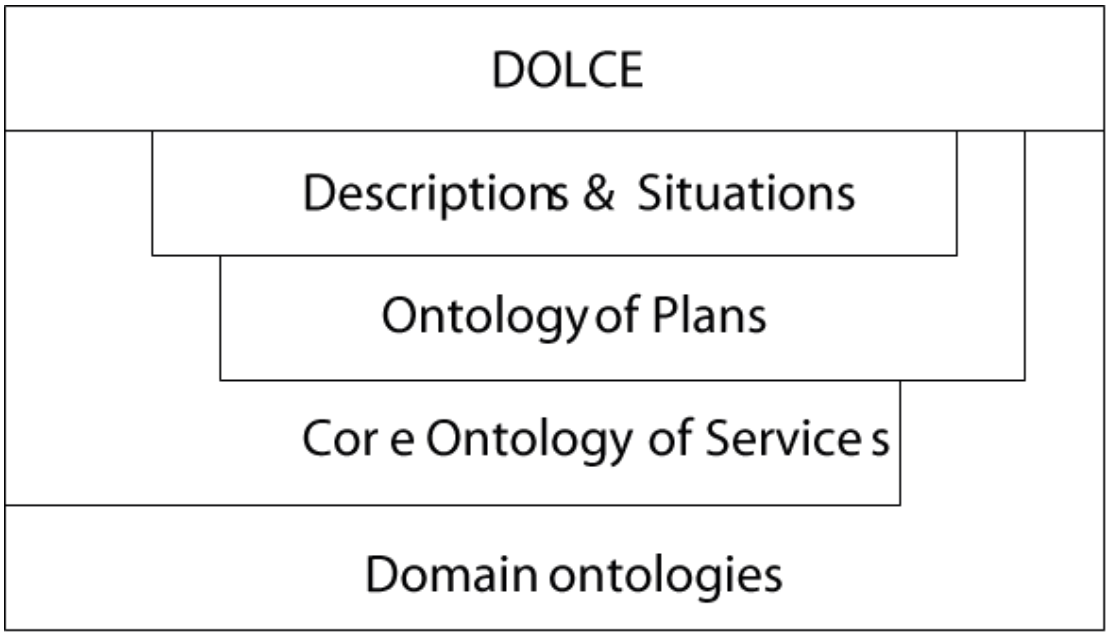

Figure 2: Core Ontology of Services.

Figure 2 shows the carefully crafted stack of ontologies we have reused and created, respectively. It consists of the DOLCE foundational ontology [11], ontology modules for context modelling (Descriptions & Situations [3]) and plans (Ontology of Plans [2]). All three are used to formulate a Core Ontology of Services[2] that models services and their interrelationships in a domain-independent way. Finally, domain ontologies specialize concepts, associations and axioms in a particular setting. For details, the reader is again referred to [9].

The role of the Core Ontology of Services with respect to existing efforts like OWL-S [13] or WSMO [1], where the intended meaning of terms is often ambiguous, is that of a well-founded least common denominator. Existing efforts can be aligned to this ontology and thus harmonized. For example, [7] shows how OWL-S can be aligned.

KAON SERVER: The Prototype

There exist several obvious target platforms for integrating our semantic management of Web Services, e.g. enterprise application management tools like IBM Tivoli, software IDEs like Eclipse, workflow engines like Microsoft's Biztalk or

Application Servers. For our prototype we have chosen to realize semantic management of Web services in an Application Server due to two reasons. First, industry-strength Web Services based applications are often realized with Application Servers as they provide the basic infrastructure. Second, we already integrated semantic technology in our own ontology-based Application Server, called KAON SERVER[3]. KAON SERVER is based on the open-source Application Server JBoss[4] and adds ontology infrastructure in order to reason with software components like EJBs or Servlets [8,10].

Our approach takes the Web service and ontology infrastructure of KAON SERVER as a basis for semantic management of Web services. Existing WS* descriptions are still fed into their corresponding engines for security, transaction or workflow. It is still necessary for the developer to familiarize and work with WS* descriptions.

However, we take WS* descriptions as a basis, i.e. we parse them, extract relevant information and integrate them as instances into our ontology. In a similar fashion, programme code and modelling tools already in use can be leveraged. As a result, the developer is able to query and reason with a harmonizing conceptual model spanning several aspects (Figure 3).

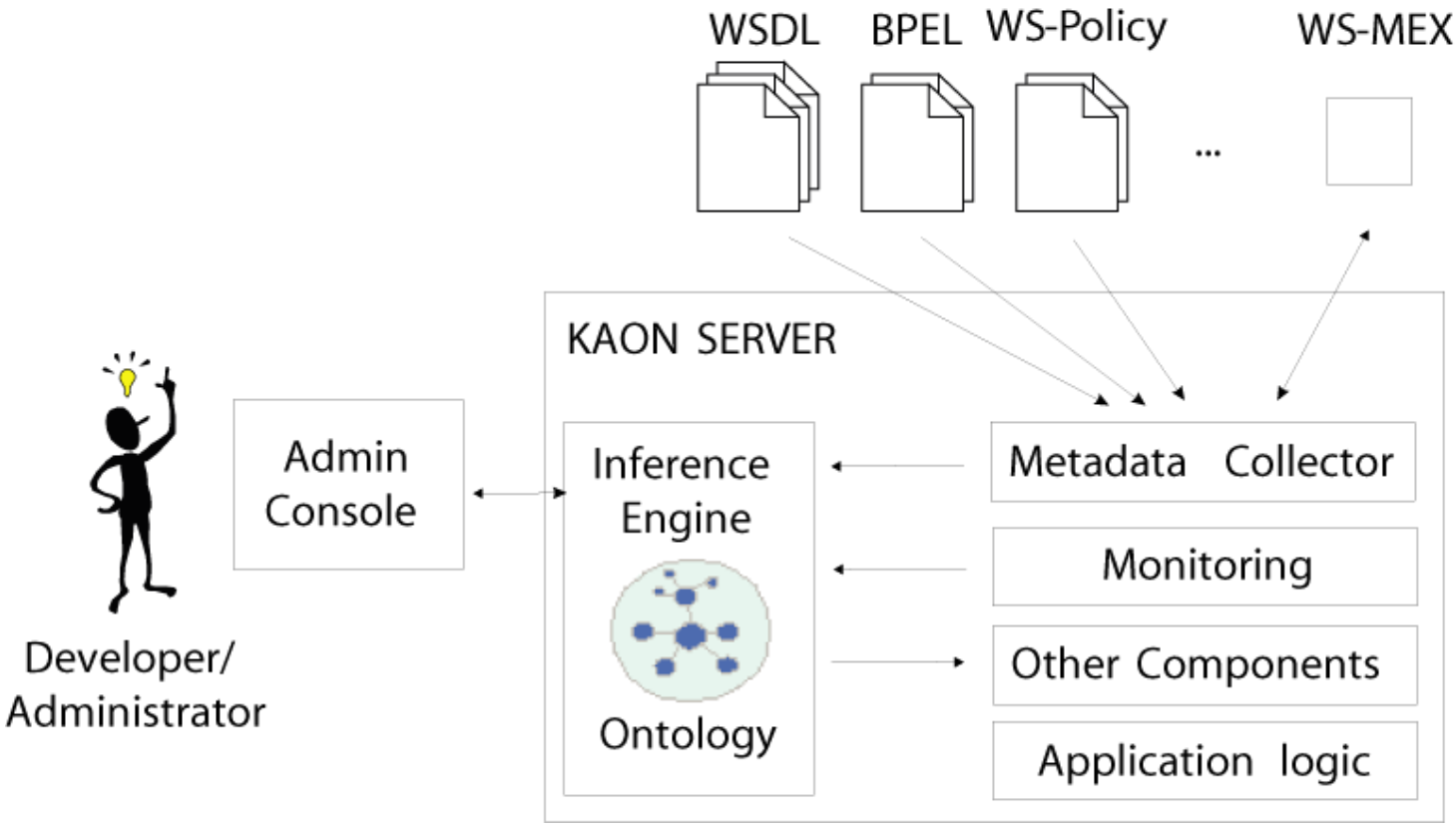

Figure 3: Integration in KAON SERVER.

An Example

As an example for a conclusion derived from both a BPEL and WS-Policy description, consider the following case. Assume a web shop realized with internal and external Web Services composed and managed by a BPEL engine. After the submission of an order, a BPEL process checks the customer's credit card for validity depending on the credit card type (VISA, MasterCard etc.). We assume that credit card providers offer this functionality via Web Services. The corresponding BPEL process `checkAccount` thus invokes one of the provider's Web Services depending on the customer's credit card.

Suppose now that the Web Service of one credit card provider, say MasterCard, only accepts authenticated invocations conforming to Kerberos or X509. It states such policies in a corresponding WS-Policy document. The invocation will fail unless the developer ensures that the policies are met. That means the developer has to check the policies manually at development

time or has to implement this functionality to react to policies at runtime.

Since process and policy information can be parsed, analyzed and integrated in our ontology, checking for the existence of external policies boils down to a simple query. Without our approach the developer would have to collect and check this information manually by analyzing BPEL and WS-Policy documents. As we may recognize from this small example, it is desirable to query a system for semantic management rather than to manually check a complex set of process definitions. We can think of more sophisticated examples where we query for particular policy constraints or where we have large indirect process cascades.

Related Efforts

Developers have to face the multitude of WS* specifications like WSDL, WS-Policy, WS-Coordination, WS-Transaction or BPEL. Because of their sheer number and disjointness, managing Web Services with WS* creates high management efforts for the developer. There is no coherent formal model of WS* and there are no means to ask for, possibly undesirable, conclusions that arise from integrating several WS* descriptions.

However, several semantic standards are arising at the moment in the field of research which is often circumscribed as ``Semantic Web Services" [6]. Unlike our approach, they aim at full automation of Web Service invocation, discovery and composition. However, common ontology languages are typically not expressive enough to reach these objectives. E.g. [6] uses a description logic which is extended by Golog to automate planning tasks. Golog is a high level programming language built on top of the situation calculus. In addition, the objective of full automation by semantic modelling will need very fine-grained, detailed modelling of all aspects of Web Services leading to high modelling costs. Our approach is located between the two extremes of WS* and semantic standards and finds a good trade-off between modelling and management costs. The following paragraph discusses some of the semantic efforts.

Two approaches try to incorporate semantic technology in UDDI, for instance. The first, [14], proposes a taxonomy support for semantics in the registry. The primary aim is to allow for a better discovery and matchmaking by leveraging the semantic descriptions. The second tries to achieve similar goals by incorporating OWL-S profiles into the UDDI registry [12]. OWL-S[5] is one of the first core ontologies explicitly aiming at automatic discovery, automatic invocation, automatic composition and interoperation as well as automatic execution of Web Services. The Web Service Modelling Ontology (WSMO) [1] has goals similar to OWL-S. However, it additionally defines an Execution Environment (WSMX) for the dynamic discovery, selection, mediation, invocation and inter-operation of Semantic Web Services. [5] take into account that most semantic efforts have been disconnected from the emerging WS* standards. Hence, they propose a ``bottom-up" approach of enriching BPEL by semantics. However, they also try to enable automated service discovery, customization, and semantic translation.

In a certain sense, our approach is similar to the way that model driven engineering (MDE) tackles a problem, viz. by abstracting modelling of some parts of an architecture, while retaining full control by the software engineer. MDE/MDA[6] and semantic management however differ, because the latter constitutes a precise, formal, executable model that may be exploited not only at compilation time (like MDE/MDA), but also for hot deployment or during runtime. Indeed, the principal idea of MDA is to separate conceptual concerns, such as which component is using which other component, from implementation-specific concerns, such as which version of an application interface requires which versions of windows libraries. MDA achieves this separation by factorizing the two concerns, specifying them separately and compiling them into an executable. Notwithstanding that MDA already provides conceptual modelling in order to improve management of complex systems, MDA is disadvantaged in two ways. First, MDA requires a compilation step preventing changes at run time which are characteristic for application server software. Second, an MDA itself cannot be queried or reasoned about. Hence, there is no way to ask the system whether some configuration is valid or whether further components are needed.

Our approach is also complementary to currently arising management specifications from the Web service community. First, the OASIS is working on Web Services Distributed Management[7]. Second, WS-Management by IBM and others is a competing

specification with similar goals. Both share some of their use case with our approach but could greatly benefit from semantic technology.

Conclusion

We have shown what semantic management of Web Services may contribute to Web Service management in general. We have described use cases for semantic management of Web Services that can be realized with existing technology and that provide immediate benefits to their target groups, i.e. software developers and administrators who deal with Web Services. The use cases shown that semantic descriptions may play a fruitful role supporting an integrated view onto Web Service definitions in WS*. At the basis of the integration we have put the Core Ontology of Services.

While we have implemented a prototype as proof-of-concept of our approach, in the long run the viability and success of semantic descriptions will only be shown in their successful use of integrated development and runtime environments. The development of the corresponding paradigm of Semantic Management of Web Services through use cases, ontologies, prototypes and examples is an important step in this direction.

Resources

A pdf version of this paper is available from the authors' webpages at http://www.aifb.uni-karlsruhe.de/WBS/dob/pubs/icws2005.pdf. The paper is based mainly on [9]. The Core Ontology of Services is available from http://cos.ontoware.org. The prototype implementation is available from http://kaon.semanticweb.org/server.

Bibliography

1

Dieter Fensel and Christoph Bussler.
The Web Service Modeling Framework WSMF.
Electronic Commerce: Research and Applications, 1:113-137, 2002.

2

Aldo Gangemi, Stefano Borgo, Carola Catenacci, and Jos Lehmann.
Task taxonomies for knowledge content.
Metokis deliverable d07, Jun 2004.

3

Aldo Gangemi and Peter Mika.
Understanding the semantic web through descriptions and situations.
In *DOA/CoopIS/ODBASE 2003 Proceedings*, LNCS. Springer, 2003.

4

Aldo Gangemi, Peter Mika, Marta Sabou, and Daniel Oberle.
An ontology of services and service descriptions.
Technical report, Laboratory for Applied Ontology (ISTC-CNR), Viale Marx, 15, 00137 Roma, 2003.

5

Daniel J. Mandell and Sheila McIlraith.
Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation.

In *2nd Int. Semantic Web Conference*, volume 2870 of *LNCS*, pages 227-247. Springer, 2003.

6

Sheila A. McIlraith, Tran Cao Son, and Honglei Zeng.
Semantic Web Services.
IEEE Intelligent Systems, 16(2):46-53, Mar 2001.

7

Peter Mika, Daniel Oberle, Aldo Gangemi, and Marta Sabou.
Foundations for Service Ontologies: Aligning OWL-S to DOLCE.
In *The 13th International World Wide Web Conference Proceedings*, pages 563-572. ACM, May 2004.

8

Daniel Oberle, Andreas Eberhart, Steffen Staab, and Raphael Volz.
Developing and managing software components in an ontology-based application server.
In *5th International Middleware Conference*, LNCS. Springer, 2004.

9

Daniel Oberle, Steffen Lamparter, Andreas Eberhart, and Steffen Staab.
Semantic management of web services.
Technical report, University of Karlsruhe, 2005.
http://www.aifb.uni-karlsruhe.de/WBS/dob/icws2005.pdf.

10

Daniel Oberle, Steffen Staab, Rudi Studer, and Raphael Volz.
Supporting Application Development in the Semantic Web.
ACM Transactions on Internet Technology (TOIT), 4(4), Nov 2004.

11

A. Oltramari, A. Gangemi, N. Guarino, and C. Masolo.
Sweetening ontologies with DOLCE.
In *Ontologies and the Semantic Web, 13th Int. Conference, EKAW 2002, Proceedings*, 2002.

12

Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia P. Sycara.
Importing the Semantic Web in UDDI.
In *CAiSE 2002 International Workshop, WES 2002*, pages 225-236, 2002.

13

The DAML Services Coalition.
OWL-S 1.0 draft release.
http://www.daml.org/services/owl-s/1.0/, Dec 2003.

14

Max Voskob.
UDDI Spec TC V4 Requirement - Taxonomy support for semantics.
OASIS, 2004.
http://www.oasis-open.org.

Footnotes

* Daniel Oberle is supported by the German Federal Ministry of Education and Research (BMBF) under the SmartWeb

project.

† Steffen Lamparter is supported by German Research Foundation in the Graduate School for Information Management and Market Engineering (DFG grant no. GRK 895).

‡ Sudhir Agarwal is supported by the German Federal Ministry of Education and Research (BMBF) under the Internetökonomie project SESAM.

§ Stephan Grimm is supported by European Union under the IST project DIP (no. FP6 - 507483).

Pascal Hitzler is supported by the German Federal Ministry of Education and Research (BMBF) under the SmartWeb project, and by the European Union under the KnowledgeWeb Network of Excellence.

[1] http://www.w3.org/2001/ws/BestPractices/SE/ODA/

[2] Maintained at http://cos.ontoware.org

[3] Available at http://kaon.semanticweb.org/server

[4] http://www.jboss.org

[5] http://www.w3.org/Submission/OWL-S/

[6] http://www.omg.org/mda/

[7] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm

---